=МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное

бюджетное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Системное программирование»

**ОТЧЕТ**

**по лабораторной работе No 1**

**дисциплина «»**

Выполнил студент гр. БВТ2201

Альвицов Д.С.

Проверил:

г. Москва

2022

Содержание

[Цель работы: 3](#__RefHeading___Toc161_1093169984)

[Задание: 3](#__RefHeading___Toc265_3210180260)

[определить проект (состоящий из нескольких файлов), который будет добавлен под версионный контроль, и произвести ряд операций в СКВ, включающих: 3](#__RefHeading___Toc267_3210180260)

[Ход работы 4](#__RefHeading___Toc167_1093169984)

[Вывод 8](#__RefHeading___Toc169_1093169984)

[Литература 9](#__RefHeading___Toc171_1093169984)

## Цель работы:

изучение базовых функций и подходов к командной разработке в системе   
контроля версий Git

## Задание:

## определить проект (состоящий из нескольких файлов), который будет добавлен под версионный контроль, и произвести ряд операций в СКВ, включающих:

1. Создание репозитория или клонирование существующего
2. Внесение ряда изменений в файл(ы) и выполнение коммитов (кол-во не меньше 2)
3. Откат изменений до одного из предыдущих коммитов двумя разными способами
4. Создание, переименование и удаление ветки
5. Слияние ветки с веткой мастер с решением конфликта содержания файлов
6. Настройка игнорирования git’ом файлов определенного расширения с   
   использованием glob-шаблонов
7. Вывод лога коммитов в измененном пользовательском формате
8. Настройка удалённого репозитория: получение и отправка изменений в   
   репозиторий
9. Организация совместной работы над проектом: клонирование чужого репозитория, выполнение коммитов, отправка изменений

# Ход работы

1) Создаю новую директорию **labs/1**, в которой инициализирую новый репозиторий командой git init

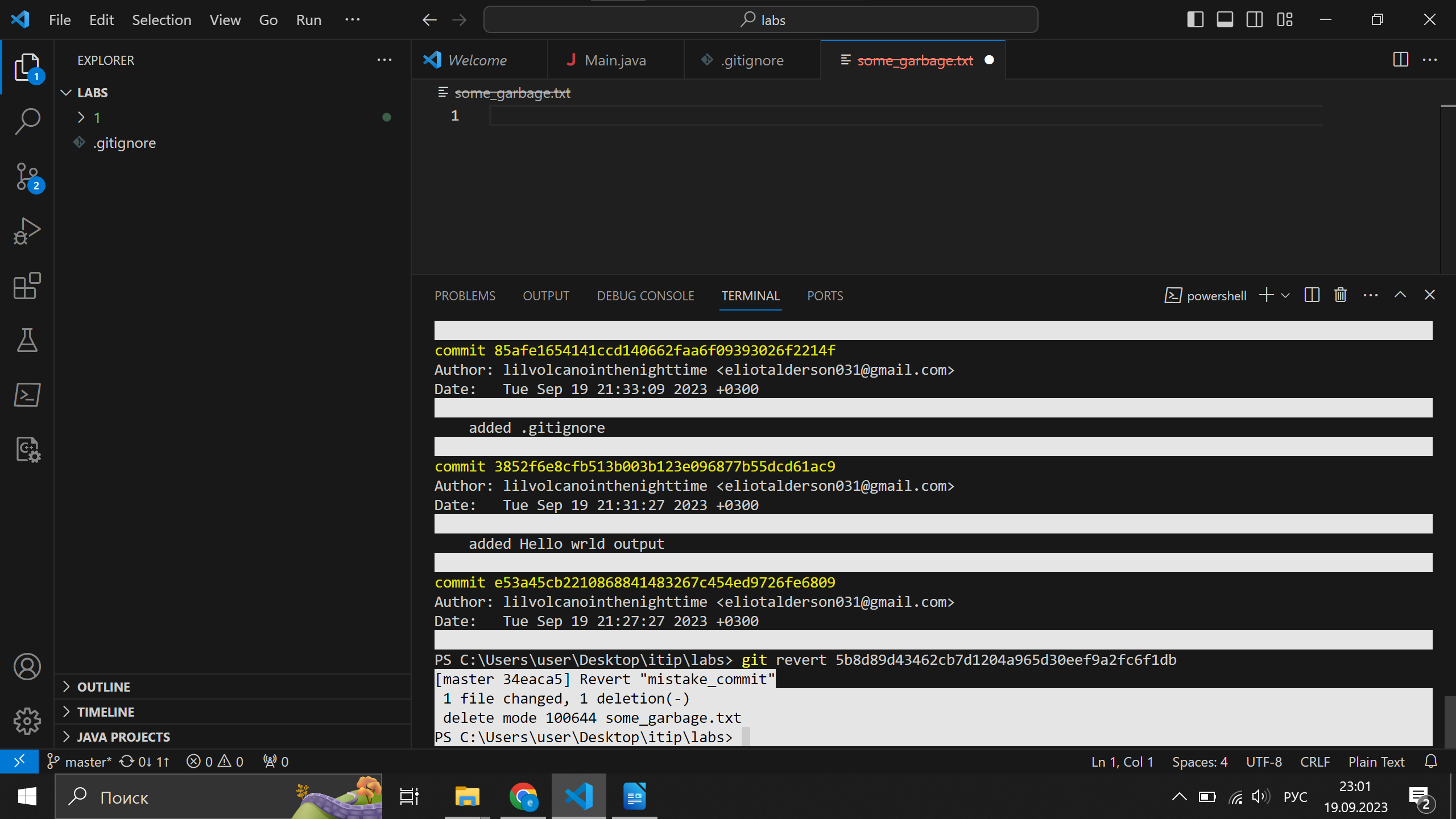
2) Делаю первый коммит с названием **first**, затем, в директории **lab/1** создаю файл **Main.java** с кодом, который выводить в консоль **Hello wrld!**, делаю коммит с названием **added Hello wrld output**, затем создаю файл **.gitignore** в директории **labs**, делаю коммит **added .gitignore**. Далее имитирую ситуацию, в которой по ошибке создаю файл **some\_garbage.txt** и делаю коммит, синхронизирую локальный и удаленный репозиторий на **github** командой

**git remote add origin** [**https://github.com/lilvolcanointhenighttime/asdasd.git**](https://github.com/lilvolcanointhenighttime/asdasd.git),

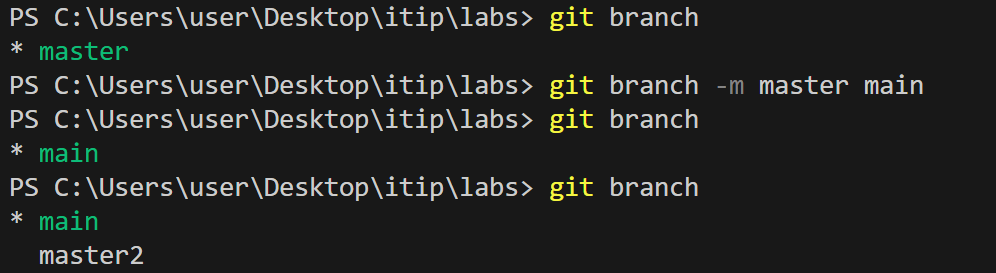
отправляю пуш в удаленный репозиторийкомандой **git push -u origin master**

3) Чтобы откатить изменения предыдущего коммита, выполняю команду

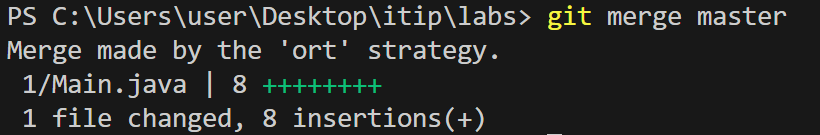
**git reset --hard HEAD~1**, далее делаю fetch с удаленного репозитория командой **git fetch**, далее выполняю команду **git merge origin/master**, чтобы сымитировать ситуацию наличия ненужного файла в репозитории, делаю комит с названием **mistake\_commit**. Удаляю ненужный файл, создавая новый коммит и не нарушая историю коммитов командой **git revert**.

Рисунок 1 – промежуточный итог выполнения работы

4)Переименовываю текущую ветку **master** командой **git branch -m master main**, создаю новую ветку командой **git branch master2**, удаляю ветку **master2** командой **git branch -d master2**.

Рисунок 2 - промежуточный итог выполнения работы

5) Создаю новую ветку **master**(старая ветка master была переименована в **main**), делаю там изменения в коде, делаю пуш, затем переключаюсь на ветку main и выполняю команду **git merge**, чтобы ветка **main** соответствовала ветке **master** в удаленном репозиториию.

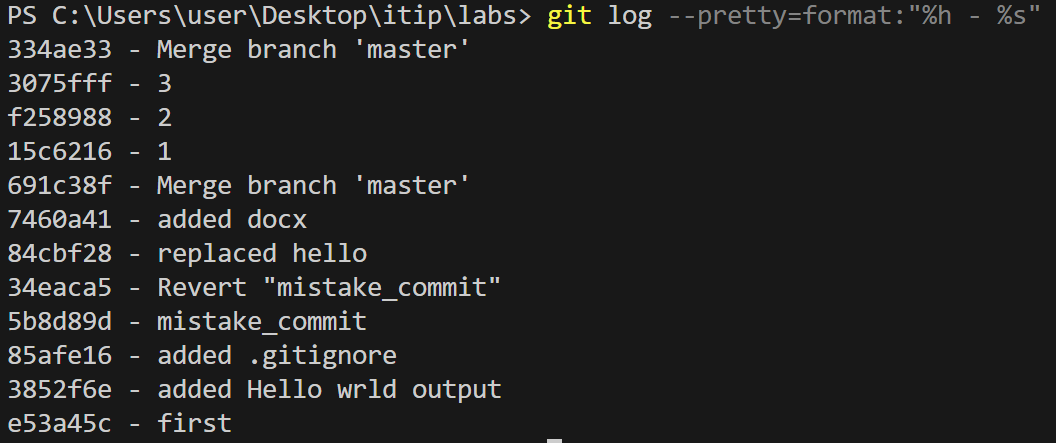
  
Рисунок 3 - промежуточный итог выполнения работы

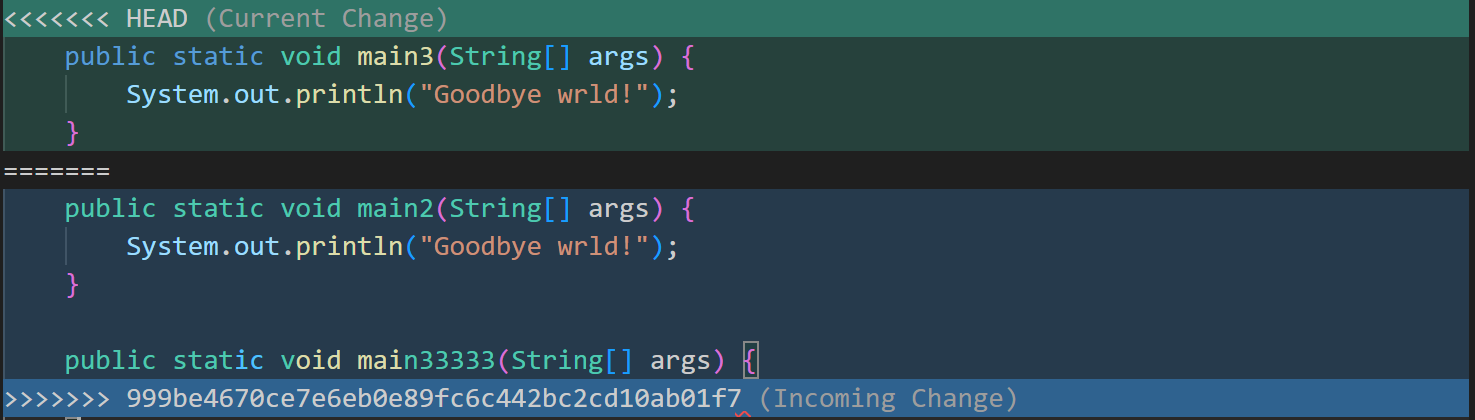
6) В созданном ранее .gitignore файле прописываю файлы, которые будут игнорироваться git’ом при выполнении команды **git add**. Использую для этого glob-шаблоны.

  
Рисунок 4 - промежуточный итог выполнения работы

7) Вывожу лог коммитов в формате «хэш-сообщение коммита» командой

**git log --pretty=format:"%h - %s"**

Рисунок 5 - промежуточный итог выполнения работы

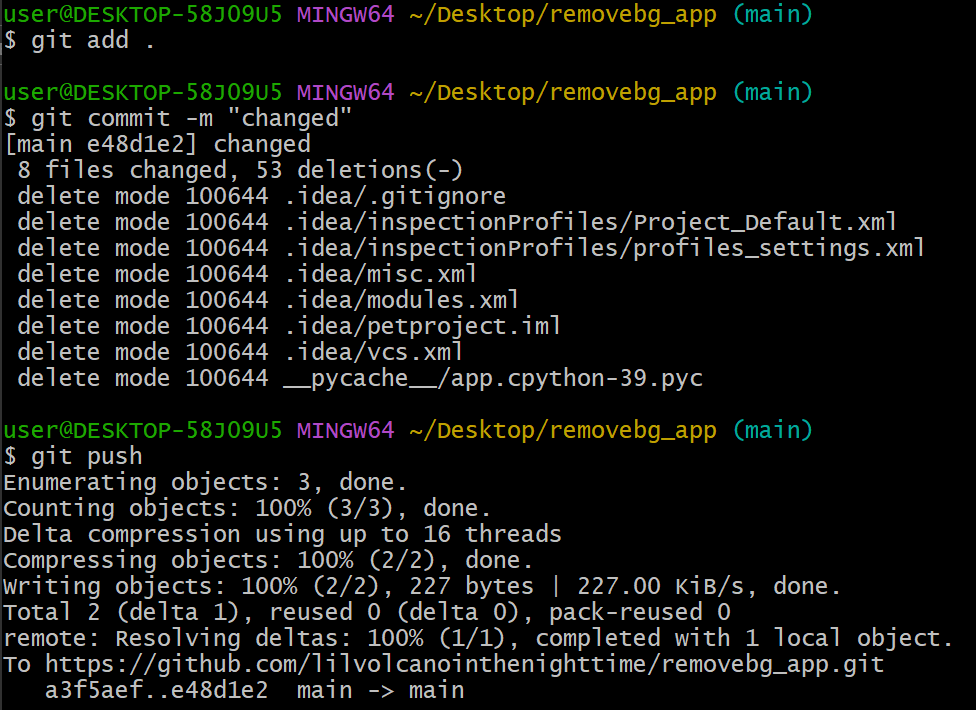
8) Делаю изменения в удаленном репозитории, копирую эти изменения с слиянием ветки командой **git pull origin master**, получаю конфликт слияния.  
  
Рисунок 6 — конфликт слияния

Переписываю код в локальном репозитории, делаю git add ./Main.java, чтобы решить конфликт слияния веток.

8) Клонирую командой **git clone** репозиторий

https://github.com/lilvolcanointhenighttime/removebg\_app.git

Удаляю директории \_\_pychache\_\_ и .idea, выполняю команду **git add .** , далее делаю коммит и выполняю пуш в удаленный репозиторий командой **git push**

  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
Рисунок 7 - клонирование чужого репозитория, выполнение коммитов, отправка изменений

# Вывод

Лабораторная работа по основам работы с системой контроля версий Git позволила овладеть важными навыками для коллективной разработки программного обеспечения. В ходе выполнения работы были изучены ключевые концепции Git, включая создание репозиториев, коммиты, ветвление и слияние. Были приобретены навыки в работе с удаленными репозиториями, а также настройки игнорирования файлов. Эти навыки будут полезны при совместной работе над проектами, управлении версиями кода и эффективной организации рабочего процесса в разработке программного обеспечения. Git оказался мощным инструментом для управления изменениями и совместной работой, и он будет широко использоваться в будущей практике разработки.

# Литература

1. Pro Git - Scott Chacon and Ben Straub